writes that are serviced as a slave to target operations. In master operations, the processor core 20 sends data directly to or receives data directly from the PCI interface 24. --

Please replace the paragraph beginning at page 8, line 22 with the following rewritten paragraph:

-- Each of the functional units are coupled to one or more internal buses. As described below, the internal buses are dual, 32 bit buses (i.e., one bus for read and one for write). The hardware-based multithreaded processor 12 also is constructed such that the sum of the bandwidths of the internal buses in the processor 12 exceeds the bandwidth of external buses coupled to the processor 12. The processor 12 includes an internal core processor bus 32, e.g., an Advanced System bus (ASB) [ASB bus (Advanced System Bus)] or AMBA bus that couples the processor core 20 to the memory controller 26a, 26[c]b and to an ASB or AMBA translator 30 described below. The ASB bus is a subset of the so called AMBA bus that is used with the Strong Arm processor core. The processor 12 also includes a private bus 34 that couples the microengine units to SRAM controller 26b, ASB translator 30 and FBUS interface 28. A memory bus 38 couples the memory controller 26a, 26b to the bus interfaces 24 and 28 and memory system 16 including flash read only memory (flashROM) 16c used for boot operations and so forth. --

Please replace the paragraph beginning at page 9, line 8 with the following rewritten paragraph: SS 8/16/07

-- Referring to FIG. 2, each of the microengines 22a-22f includes an arbiter that examines flags to determine the available threads to be operated upon. Any thread from any of the microengines 22a-22f can access the SDRAM controller 26a, [SDRAM] SRAM controller 26b or FBUS interface 28. The memory controllers 26a and 26b each include a plurality of queues to store outstanding memory reference requests. The queues either maintain order of memory

Please replace the paragraph beginning at page 24, line 31 with the following rewritten paragraph:

-- A second one is that microcontroller or microengine execution latency should be optimized at the expense of overall microengine compute throughput and overall memory bandwidth. This paradigm could involve execution of a thread with a real-time constraint, that is, a constraint which dictates that some work must absolutely be done by some specified time. Such a constraint requires that optimization of the single thread execution be given priority over other considerations such as memory bandwidth or overall computational throughput. A real-time thread would imply a single microengine that executes only one thread. Multiple threads would not be handled because the goal is to allow the single real-time thread to execute as soon as possible--execution of multiple threads would hinder this ability. --

Please replace the paragraph beginning at page 25, line 13 with the following rewritten paragraph: SS 8/16/07

-- The coding style of these two paradigms could be significantly different with regard to issuing memory references and context switching. In the real time case, the goal is to issue as many memory references as soon as possible in order to minimize the memory latency incurred by those references. Having issued as many references as early as possible the goal would be to perform as many computations [as] in the microengines as possible in parallel with the references. A computation flow that corresponds to real-time optimization is: --

Please replace the paragraph beginning at page 26, line 15 with the following rewritten paragraph:

processor 20 and the PCI interface 24. The SDRAM controller 26a also maintains a state machine for performing READ-MODIFY-Write atomic operations. The SDRAM controller 26a also performs byte alignment for requests of data from SDRAM. --

Please replace the paragraph beginning at page 30, line 28 with the following rewritten paragraph: SS 8/14/07

-- The SDRAM controller 26a also includes core bus interface logic [i.e., ASB bus 92]. The ASB bus interface logic 92 interfaces the core processor 20 to the SDRAM controller 26a. The ASB bus is a bus that includes a 32 bit data path and a 28 bit address path. The data is accessed to and from memory through MEM ASB data device 98, e.g., a buffer. MEM ASB data device 98 is a queue for write data. If there is incoming data from the core processor 20 via ASB interface logic 92, the data can be stored into the MEM ASB device 98 and subsequently removed from MEM ASB device 98 through the SDRAM interface 110 to SDRAM memory 16a. Although not shown, the same queue structure can be provided for the reads. The SDRAM controller 26a also includes an engine 97 to pull data from the microengines and PCI bus. --

Please replace the paragraph beginning at page 30, line 1 with the following rewritten paragraph:

-- The order queue 90c maintains the order of reference requests from the microengines. With a series of odd and even bank[s] references it may be required that a signal is returned only upon completion of a sequence of memory references to both the odd and even banks. If the microengine 22f sorts the memory references into odd bank and even bank references and one of the banks, e.g., the even bank is drained of memory references before the odd bank but the signal is asserted on the last even reference, the memory controller 26a could conceivably signal back to a microengine that the memory request had completed, even though the odd bank reference